

An efficient approximation for point set diameter in higher dimensions

Mahdi Imanparast and Seyed Naser Hashemi *

Department of Computer Science, Faculty of Mathematics and
Computer Science, Amirkabir University of Technology,
Tehran, Iran
`m.imanparast@aut.ac.ir, nhashemi@aut.ac.ir`

Abstract. In this paper, we study the problem of computing the diameter of a set of n points in d -dimensional euclidean space, for a fixed dimension d , and propose a new $(1 + \varepsilon)$ -approximation algorithm with $O(n + 1/\varepsilon^{d-2})$ time and $O(n)$ space, where $0 < \varepsilon \leq 1$. The proposed algorithm is simple in implementation and does not need complicated data structures. We also improve the proposed algorithm to another algorithm with $O(n + 1/\varepsilon^{(d-1)/2})$ running time.

Keywords: Diameter, point sets, approximation algorithm, higher dimensions.

1 Introduction

Given a finite set \mathcal{S} of n points, the diameter of \mathcal{S} denoted by $D(\mathcal{S})$ is the maximum distance between two points of \mathcal{S} . Computing the diameter of a set of points has a large history and it may be required in various fields such as database, data mining and vision. A trivial algorithm for this problem is as follows: compute the distance between each pair of points and then choose the maximum distance. Since, computing the distance takes constant time, this algorithm takes $O(n^2)$ time. But, this is too slow for large-scale data sets that occur in the fields. Hence, we need a faster algorithm which may be exact or approximation.

By reduction to set disjointness problem, it can be shown that computing the diameter of n points in \mathbb{R}^d requires $\Omega(n \log n)$ operations in algebraic computation-tree model [1]. It is shown by Yao that, it is possible to compute the diameter in sub-quadratic time in each dimension [2]. In two and three dimensions, there are well-known solutions. In the plane, this problem can be computed in optimal time $O(n \log n)$, but, in three dimensions, it is hard. Clarkson and Shor [3] present an $O(n \log n)$ -time randomized algorithm. Their algorithm needs to compute the intersection of n balls (with the same radius) in \mathbb{R}^3 . It may be slower than the brute-force algorithm for the most practical data sets. Moreover, It is not an efficient method for higher dimensions, because the intersection of n balls with the same radius has a large size. Some deterministic algorithms with

* Corresponding Author.

running time $O(n \log^3 n)$ [4,5] and $O(n \log^2 n)$ [6,7] are found for solving this problem. Finally, Ramos [8,9] found an optimal deterministic $O(n \log n)$ -time algorithm in \mathbb{R}^3 . Cheong et.al [10] present an $O(n^2 \log n)$ randomized algorithm that computes the all-pairs farthest neighbors for n points on the convex position in \mathbb{R}^3 .

In the absence of fast algorithms, many attempts have been done to approximate the diameter in low and high dimensions. A 2-approximation algorithm with $O(dn)$ time in d dimensions can be found easily by selecting a point x of \mathcal{S} and then finding the farthest point y from it by brute-force manner [11]. The first non-trivial approximation algorithm for this problem is presented by Egecioglu and Kalantari [12] that approximates the diameter with factor $\sqrt{3}$ and operations cost $O(dn)$ in d dimensions. Also, they present an iteration algorithm with $t \leq n$ iterations and the cost $O(dn)$ for each iteration that has approximate factor $\sqrt{5 - 2\sqrt{3}}$.

Agarwal et. al [13] present a $(1 + \varepsilon)$ -approximation algorithm in \mathbb{R}^d with $O(n/\varepsilon^{(d-1)/2})$ running time by projecting to directions. Barequet and Har Peled [14] present a \sqrt{d} -approximate diameter method with $O(dn)$ time, since d is fixed, it can be denoted by $O(n)$. They also describe a $(1 + \varepsilon)$ -approximate approach for computing the diameter \hat{D} such that $\hat{D} \leq D \leq (1 + \varepsilon)\hat{D}$ with $O(n + 1/\varepsilon^{2d})$ time in \mathbb{R}^d . They show that, the running time can be improved to $O(n + 1/\varepsilon^{2(d-1)})$. Similarly, Chan [16] observes that a combination of two approaches in [13] and [14] yields an $(1 + \varepsilon)$ -approximate algorithm with $O(n + 1/\varepsilon^{3(d-1)/2})$ and an $(1 + O(\varepsilon))$ -approximate algorithm with $O(n + 1/\varepsilon^{d-1/2})$ time. He also introduces a core-set theorem, and shows that using this theorem, an $(1 + O(\varepsilon))$ -approximation for the diameter in $O(n + 1/\varepsilon^{d-3/2})$ time can be found [17].

Also, Har Peled [15] present an approach which for the most inputs, it is able to compute the exact diameter or an approximation very fast. Although, in the worst case, the algorithm running time is still quadratic, and is sensitive to the hardness of the input. His algorithm is able to return a pair of points p and q such that $|p - q| \geq (1 - \varepsilon)D$, for each value $\varepsilon > 0$ in each dimension, with $O((n + 1/\varepsilon^{2d}) \log 1/\varepsilon)$ running time. He shows that with a complicated analysis, this running time can be reduced to $O((n + 1/\varepsilon^{3(d-1)/2}) \log 1/\varepsilon)$. Simultaneously, Maladain and Boissonnat [18] present an exact algorithm for the diameter by computing the double normals in each dimension. But, their algorithm is not worst-case optimal. They also show, with having double normals, a $\sqrt{3}$ -approximation of the diameter in each dimension is provided. Moreover, Finocchi and Pellegrini [19] describe an algorithm that finds in $O(dn \log n + n^2)$ time with high probability a $(1 + \varepsilon)$ -approximation of the diameter of a set of n points in d -dimensional euclidean space.

1.1 Related works

As previously mentioned, some $(1 + O(\varepsilon))$ -approximation algorithms have been presented in literatures. The first result is an $(1 + \varepsilon)$ -approximate with running time $O(n/\varepsilon^{(d-1)/2})$ due to Agarwal et. al [13] which is based on projection to

directions. Other methods which present approximate approaches with running time $O(n + c_d \cdot (1/\varepsilon^{O(d)}))$, where they have usually a hidden constant c_d with exponential dependency on d . They include Barequet and Har Peled [14] method, Chan's [16,17] results which are a combination of two approaches in [13] and [14], and finally, Har Peled [15] approach. However, most of these methods use a similar idea (except the last one). They first compute a 2-approximation Δ_0 of the diameter of the point set S in $O(dn)$ time by choosing a point x of S , and then finding its farthest point y , and use Δ_0 for constructing a grid. Then, round each point to its nearest grid-point and find the diameter in this new point set. But, using this idea has the following properties. Firstly, it leads to a rounded point set with $(16\sqrt{d})^d/\varepsilon^d$ points (balls) which needs $((16\sqrt{d})^d/\varepsilon^d)^2$ time to compute its diameter. Secondly, it uses a 2-approximation, which chooses a point randomly. This means that, if we run these algorithms twice on an input set, they might result different value for the approximate diameter. Thus, the precisely running time of Barequet and Har Peled [14] method is $O(n + (16\sqrt{d})^{2d}/\varepsilon^{2d})$ [11]. Since, d is fixed, they assume that $O((16\sqrt{d})^d/\varepsilon^d) = O(1/\varepsilon^d)$ and then result the running time $O(n + 1/\varepsilon^{2d})$. This condition also holds for Chan's results [16,17]. Finally, Har Peled [15] applied a different approach which is used a fair-split tree. His algorithms running time precisely is $O((n + (4d)^{2d}/\varepsilon^{2d}) \log 1/\varepsilon)$. By assuming that $O((4d)^d/\varepsilon^d) = O(1/\varepsilon^d)$, he obtains the running time $O((n + 1/\varepsilon^{2d}) \log 1/\varepsilon)$.

1.2 Our results

In this paper, we propose a new $(1 + \varepsilon)$ -approximation algorithm for computing the diameter of a set S of n points in \mathbb{R}^d with $O(n + 1/\varepsilon^{d-2})$ time and $O(n)$ space, where $0 < \varepsilon \leq 1$. Our proposed algorithm obtains new results by applying a different approach in compare with the previous methods. Firstly, we use a constant value ℓ for constructing our grids instead of using the 2-approximate diameter Δ_0 . The value ℓ is the largest side of the axis-parallel bounding box of the point set. So, our algorithm produces same results, if we run it twice or more, because, it depends on a constant value ℓ . Secondly, we round each points to its central-cell points in a grid which helps us to get a better running time by decreasing the search domain. Moreover, we propose an approximate algorithm with $O(n + 1/\varepsilon^{(d-1)/2})$ time. So, we obtain some approximation algorithms which have some improvements in the running time in compare with previous algorithms for computing the diameter in fixed dimensions. The reminder of this paper is organized as follows: in section 2, we describe our proposed algorithm. Subsection 2.1 includes our analysis over the algorithm. Section 3 is conclusions.

2 Algorithm

In this subsection, we describe our new $(1 + \varepsilon)$ -approximate algorithm to compute the diameter of a set S of n points in \mathbb{R}^d . In this problem, we want find a diametrical pair p_D and q_D such that $D(S) = |p_D - q_D| = \max_{p,q \in S} (|p - q|)$. To this end, we first find extreme points in each coordinate and compute the

axis-parallel bounding box of \mathcal{S} , which is denoted by $B(\mathcal{S})$. We use the largest length side of the $B(\mathcal{S})$ to impose two grids on it. In fact, we decompose $B(\mathcal{S})$ to a grid of regular hypercubes with side length ϵ , where $\epsilon = O(\varepsilon\ell)$. We call each hypercube as a cell. Then, each point of \mathcal{S} is rounded to its corresponding central cell-point. Fig 1 shows an example of the rounding process for a point set in \mathbb{R}^2 .

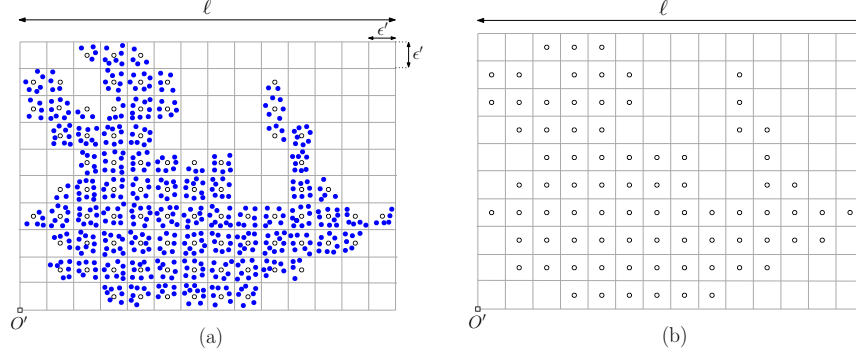


Fig. 1. (a) A set of points in \mathbb{R}^2 and an ϵ -grid. Initial points are shown by blue points and their corresponding central cell-points are shown by circle points. (b) Rounded point set $\hat{\mathcal{S}}$.

As can be seen in Fig 1, many points may be rounded to a point in $\hat{\mathcal{S}}$. But, for improving the running time of the algorithm, in this stage, we impose an ϵ_1 -grid to $B(\mathcal{S})$, for $\epsilon_1 = O(\sqrt{\varepsilon}\ell)$ and round each point of the new point set $\hat{\mathcal{S}}$ to its central cell-point in this new grid. This results a new rounded point set $\hat{\mathcal{S}}_1$. Note that, we only need $O(dn)$ time to round points to its central-cell points. Let, $B_{\epsilon_1}(p)$ be a hypercube with side length ϵ_1 and central-point p . This step reduces regions which may include the diametrical pairs to two hypercubes $B_{\epsilon_1}(p_1)$ and $B_{\epsilon_1}(q_1)$, and helps us to give a better running time. Let, two point sets B_1 and B_2 include points of the rounded point set $\hat{\mathcal{S}}$, which are inside two hypercubes $B_{\epsilon_1}(p_1)$ and $B_{\epsilon_1}(q_1)$, respectively. It is sufficient to find diameter between points of $\hat{\mathcal{S}}$, which are in two sets B_1 and B_2 .

2.1 Analysis

In this subsection, we analysis the proposed algorithm.

Theorem 1. *The proposed approximate algorithm computes the diameter of a set \mathcal{S} of n points in \mathbb{R}^d in $O(n + 1/\varepsilon^{d-2})$ time and $O(n)$ space, where $0 < \varepsilon \leq 1$.*

Proof. Finding the extreme points in all coordinate and finding the largest side of $B(\mathcal{S})$ can be done in $O(dn)$ time. Rounding step takes $O(d)$ time for each point and for all of them takes $O(dn)$ time. Since, d is fixed, this can be shown by $O(n)$. But, for computing the approximate diameter over the rounded point

set \hat{S}_1 , we need to know the number of points in the set \hat{S}_1 . We know that the largest side of the bounding box $B(\mathcal{S})$ has length ℓ and the side length of each cell is $\epsilon_1 = (\sqrt{\epsilon}\ell/2\sqrt{d})$. On the other hand, the volume of a hypercube of side length L in d -dimensional space is L^d , therefore, the number of central cell-points in the bounding box $B(\mathcal{S})$ is at most $O(2\sqrt{d}/\epsilon^{d/2})$. So, the number of points in \hat{S}_1 is at most $O(1/\epsilon^{d/2})$. Hence, by brute-force we need $O((1/\epsilon^{d/2})^2) = O(1/\epsilon^d)$ time for computing the diameter of rounded point set \hat{S}_1 .

In the next step of the algorithm, we need to compute two sets B_1 and B_2 , which include points of \hat{S} are in two hypercubes $B_{\epsilon_1}(p_1)$ and $B_{\epsilon_1}(q_1)$, respectively. Clearly, this work takes $O(dn)$ time. Then, for finding the diameter between two sets B_1 and B_2 , we need to know number of points on each of them. On the other hand, maximum number of points in two sets B_1 and B_2 is $O(1/\epsilon^{d/2})$. So, for computing $\text{diam}(B_1, B_2)$, we need to $O((1/\epsilon^{d/2})^2) = O(1/\epsilon^d)$ time. But, we may have more than one diametrical pairs B_1 and B_2 . Since, we may have at most 2^{d-1} different diametrical pairs B_1 and B_2 , this step takes at most $O(2^{d-1}/\epsilon^d)$ time. So, total complexity of our algorithm is as follows:

$$O(n + (2\sqrt{d})^2/\epsilon^d). \quad (1)$$

Since, d is fixed, we have: $O(n + 1/\epsilon^d)$.

We can reduce running time of the proposed algorithm by discarding some central cell-points which do not have any potential to be the diametrical pairs in rounded point set \hat{S}_1 , and similarly, in two point sets B_1 and B_2 . If we consider all the points which are same in their first $(d-1)$ coordinates, and keep only highest and lowest. Then, number of points in \hat{S}_1 , and two point sets B_1 and B_2 can be reduced to $O(1/\epsilon^{d/2-1})$. So, using the brute-force quadratic algorithm we need $O((1/\epsilon^{d/2-1})^2)$ time to find the diametrical pair. Hence, this gives us the total running time $O(n + 1/\epsilon^{d-2})$. About required space, we only need to $O(n)$ space for storing required points sets. So, this completes the proof. \square

Our last result is based on a combination of two approaches which are used in [13] and [16]. Let, we round initial point set and obtain a reduced point set with $O(1/\epsilon^{d/2-1})$ points in $O(n)$ time. After rounding step, in the Agrawal et. al [13] method instead of projecting the points on $O(1/\epsilon^{(d-1)/2})$ different directions which need many projection operations, we project points on some hyperplanes and compute the approximate diameter in these hyperplanes. This results in an improved approximation that we have presented in the following Corollary.

Corollary 2. *An $(1 + O(\epsilon))$ -approximate for the diameter of a set of n points in d -dimensional euclidean space, can be computed in $O(n + 1/\epsilon^{(d-1)/2})$ time, where $0 < \epsilon \leq 1$.*

Now, back to the details of the approximation. In the next lemma, we show that the proposed algorithm computes an approximate diameter \tilde{D} with error range ϵD from the true diameter D , where $0 < \epsilon \leq 1$.

Lemma 3. *Algorithm 1 computes an approximate diameter \tilde{D} such that: $D \leq \tilde{D} \leq (1 + \epsilon)D$, where $0 < \epsilon \leq 1$.*

Proof. Let \hat{p} and \hat{q} be two central cell-points that are used by the algorithm for computing the approximate diameter \hat{D} . Then, we have two corresponding exact points p and q for these two approximate points. Fig 2 illustrates a rounded point set and its diameter.

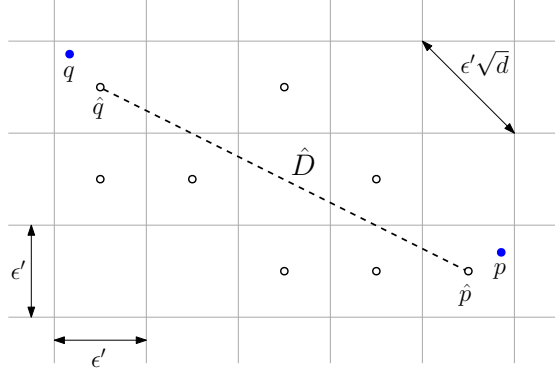


Fig. 2. Approximate diameter \hat{D} is computed over rounded point set $\hat{\mathcal{S}}$ in \mathbb{R}^2 . Two central cell-points \hat{p} and \hat{q} are used to compute \hat{D} . Their corresponding initial points are p and q .

So, we have:

$$D - \epsilon\sqrt{d} \leq \hat{D} \leq D + \epsilon\sqrt{d}. \quad (2)$$

Now, we can simplify (2) as following:

$$D \leq \hat{D} + \epsilon\sqrt{d} \leq D + 2\epsilon\sqrt{d}. \quad (3)$$

By setting $\epsilon = O(\varepsilon\ell)$, we have:

$$D \leq \hat{D} + \varepsilon\ell/2 \leq D + \varepsilon\ell. \quad (4)$$

Since $\ell \leq D$, and by setting $\tilde{D} = \hat{D} + \varepsilon\ell/2$, we can result:

$$D \leq \tilde{D} \leq (1 + \varepsilon)D. \quad (5)$$

Therefore, the lemma is resulted. \square

3 Conclusions

We have presented an efficient $(1 + \varepsilon)$ -approximation algorithm to compute the diameter of a point set \mathcal{S} in \mathbb{R}^d , for a fixed dimension d in $O(n + 1/\varepsilon^{d-2})$ time, when $0 < \varepsilon \leq 1$. The proposed algorithm used new approach that results a method which are simple in implementation. Moreover, we show that the running time of the proposed algorithm can be improved to an $O(1 + O(\varepsilon))$ -approximate with $O(n + 1/\varepsilon^{(d-1)/2})$ time. Thus, these results provide some improvements in the running time of this problem.

References

1. Preparata, F.P., Shamos, M.I.: Computational Geometry: an Introduction. New York, Springer-Verlag, pp. 176-182, (1985)
2. Yao, A. C.: On constructing minimum spanning trees in k -dimensional spaces and related problems. SIAM Journal of Computing, 11, pp. 721-736, (1982)
3. Clarkson, K. L., Shor, P. W.: Applications of random sampling in computational geometry. Discrete and Computational Geometry, 4, pp. 387-421, (1989)
4. Amato, N. M., Goodrich, M. T., Ramos, E. A.: Parallel algorithms for higher dimensional convex hulls. In Proceedings of 35th Annual Symposium on Foundations of Computer Science, pp. 683-694, (1994)
5. Ramos, E. A.: Intersection of unit-balls and diameter of a point set in \mathbb{R}^3 . Computational Geometry: Theory and Applications, 8, pp. 57-65, (1997)
6. Ramos, E. A.: Construction of 1-d lower envelopes and applications. In Proceedings of 13th Annual ACM Symposium Computational Geometry, pp. 57-66, (1997)
7. Bespamyatnikh, S.: An efficient algorithm for the three-dimensional diameter problem. Discrete and Computational Geometry, 25(2), pp. 235-255, (2001)
8. Ramos, E. A.: Deterministic algorithms for 3-D diameter and some 2-D lower envelopes. In Proceedings of the sixteenth annual symposium on Computational Geometry, (2000)
9. Ramos E. A.: An optimal deterministic algorithm for computing the diameter of a three-dimensional point set. Discrete and Computational Geometry, 26, pp. 245-265, (2001)
10. Cheong, O, Shin, C. S., Vigneron A.: Computing farthest neighbors on a convex polytope. In Proceedings of the 7th Annual International Computational and Combinatoric Conference, volume 2108 of LNCS, pp. 159-169, (2001)
11. Har-Peled, S.: Geometric Approximation Algorithms (Mathematical Surveys and Monographs). American Mathematical Society Press, (2011)
12. Egecioglu, O., Kalantari B.: Approximating the diameter of a set of points in the Euclidean space. Information Processing Letters, 32(4), pp. 205-211, (1989)
13. Agarwal, P.K., Matousek, J. , Suri, S.: Farthest neighbors maximum spanning trees and related problems in higher dimensions. Computational Geometry: Theory and Applications, 1, pp. 189-201, (1992)
14. Barequet, G., Har-Peled, S.: Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. Journal of Algorithms, 38, pp. 91-109, (2001)
15. Har-Peled, S.: A practical approach for computing the diameter of a point set. In Proceedings of the 17th annual symposium on Computational Geometry (SOCG'2001), pp. 177-186, (2001)
16. Chan, T. M.: Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. International Journal of Computational Geometry and Applications, pp. 67-85, (2002)
17. Chan, T. M.: Faster core-set constructions and data stream algorithms in fixed dimensions, Computational Geometry: Theory and Applications, 35 (1-2), pp. 20-35, (2006)

18. Malandain, G. , Boissonnat, J. D.: Computing the diameter of a point set. International Journal of Computational Geometry and Applications, 12(6), pp. 489-509, (2002)
19. Finocchiaro, D. V., Pellegrini, M.: On computing the diameter of a point set in high dimensional Euclidean space. Theoretical Computer Science, 287, pp. 501-514, (2002)